



9. More Bioconductor packages

Thomas Lumley
Ken Rice

Universities of Washington and Auckland

Seattle, July 2013

GWAS analysis

Genome-Wide Association Studies (GWAS) are currently popular – typically, these genotype e.g. 1M SNPs on several thousand subjects in (large) established studies

- Usually on 1000's of subjects
- 'Simple' t -tests, regressions, for each SNP (like microarrays)
- 1M *anything* takes a long time! (up to 72 hours)
- Just **loading** big datasets is non-trivial – but some tools are available

GWAS analysis

`snpStats` is a Bioconductor package for GWAS analysis – maintained by David Clayton (analysis lead on Wellcome Trust)

```
> biocLite("snpStats") #in a new session
> library(snpStats)
> data(for.exercise)
> ls()
[1] "snp.support" "snps.10" "subject.support"
```

A ‘little’ case-control dataset (Chr 10) based on HapMap – three objects; `snp.support`, `subject.support` and `snps.10`

GWAS analysis

```
> summary(snp.support)
  chromosome      position      A1      A2
Min.      :10   Min.      : 101955  A:14019  C: 2349
1st Qu.   :10   1st Qu.   : 28981867  C:12166  G:12254
Median    :10   Median    : 67409719  G: 2316  T:13898
Mean      :10   Mean      : 66874497
3rd Qu.   :10   3rd Qu.   :101966491
Max.      :10   Max.      :135323432

> summary(subject.support)
      cc      stratum
Min.    :0.0   CEU      :494
1st Qu. :0.0   JPT+CHB:506
Median  :0.5
Mean    :0.5
3rd Qu. :1.0
Max.    :1.0
```

GWAS analysis

```
> show(snps.10) # show() is generic
A SnpMatrix with 1000 rows and 28501 columns
Row names:  jpt.869 ... ceu.464
Col names:  rs7909677 ... rs12218790
> summary(snps.10)
$rows
  Call.rate      Heterozygosity
Min.   :0.9879   Min.   :0.0000
Median :0.9900   Median :0.3078
Mean   :0.9900   Mean   :0.3074
Max.   :0.9919   Max.   :0.3386
$cols
  Calls      Call.rate      MAF      P.AA
Min.   : 975   Min.   :0.975   Min.   :0.0000   Min.   :0.00000
Median : 990   Median :0.990   Median :0.2315   Median :0.26876
Mean   : 990   Mean   :0.990   Mean   :0.2424   Mean   :0.34617
Max.   :1000   Max.   :1.000   Max.   :0.5000   Max.   :1.00000
  P.AB      P.BB      z.HWE
Min.   :0.0000   Min.   :0.00000   Min.   : -21.9725
Median :0.3198   Median :0.27492   Median :  -1.1910
Mean   :0.3074   Mean   :0.34647   Mean   :  -1.8610
Max.   :0.5504   Max.   :1.00000   Max.   :   3.7085
                        NA's   :   4.0000
```

GWAS analysis

- 28501 SNPs, all with Allele 1, Allele 2
- 1000 subjects, 500 controls (`cc=0`) and 500 cases (`cc=1`)
- **Far too much** data for a regular `summary()` of `snps.10` – even in this small example

GWAS analysis

We'll use just the column summaries, and a (mildly) 'clean' subset;

```
> snpsum <- col.summary(snps.10)
> use <- with(snpsum, MAF > 0.01 & z.HWE^2 < 200)
```

```
> table(use)
```

```
use
```

```
FALSE TRUE
```

```
317 28184
```

GWAS analysis

Now do single-SNP tests for each SNP, and extract the p -value for each SNP, along with its location;

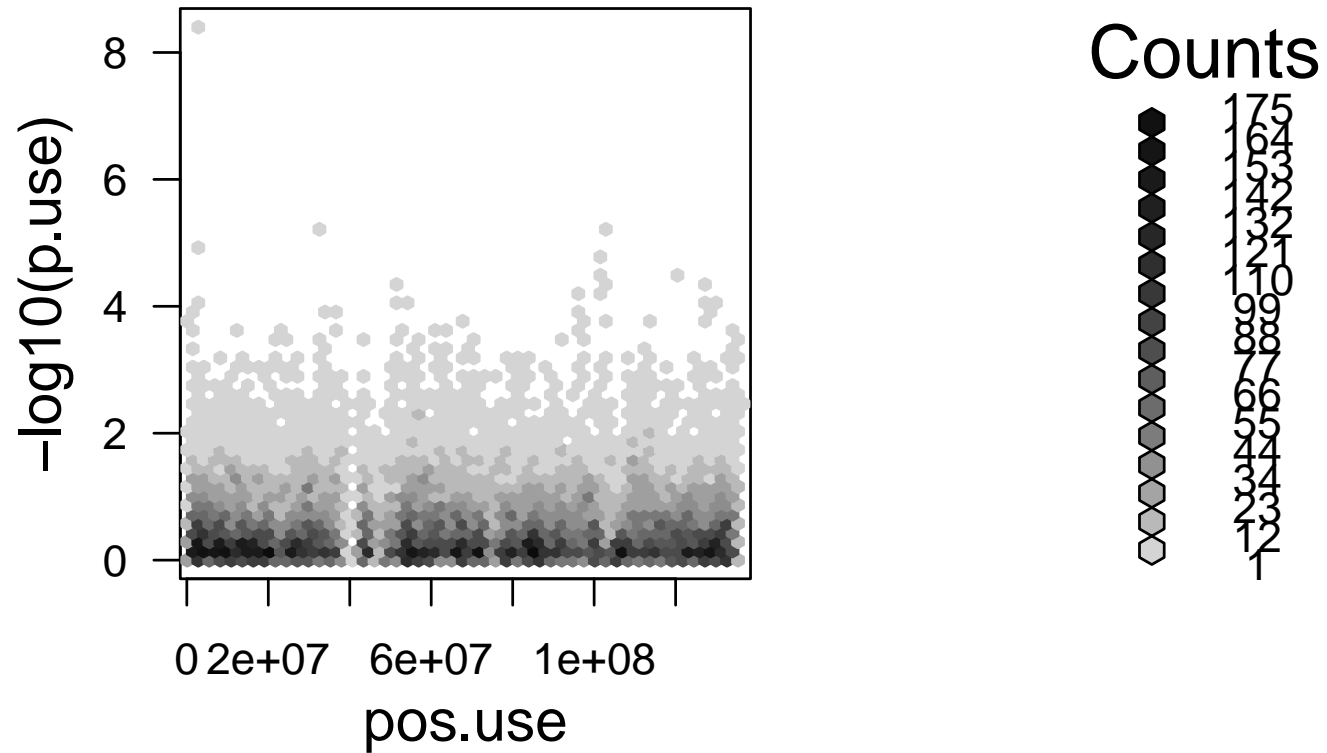
```
tests <- single.snp.tests(cc, data = subject.support,  
+ snp.data = snps.10)
```

```
pos.use <- snp.support$position[use]  
p.use   <- p.value(tests, df=1)[use]
```

We'd usually give a table of 'top hits,' but...

GWAS analysis

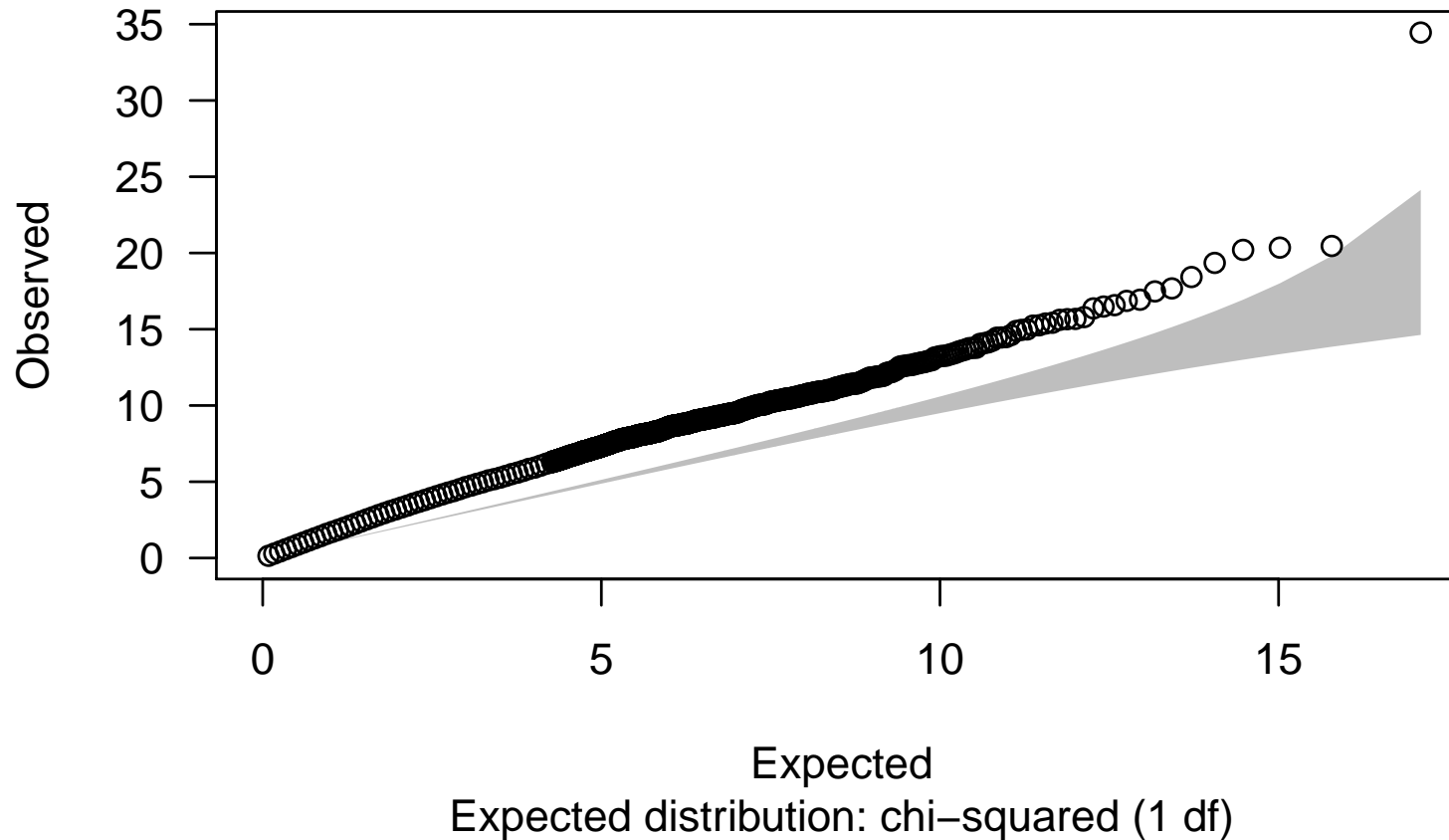
```
plot(hexbin(pos.use, -log10(p.use), xbin = 50))
```



GWAS analysis

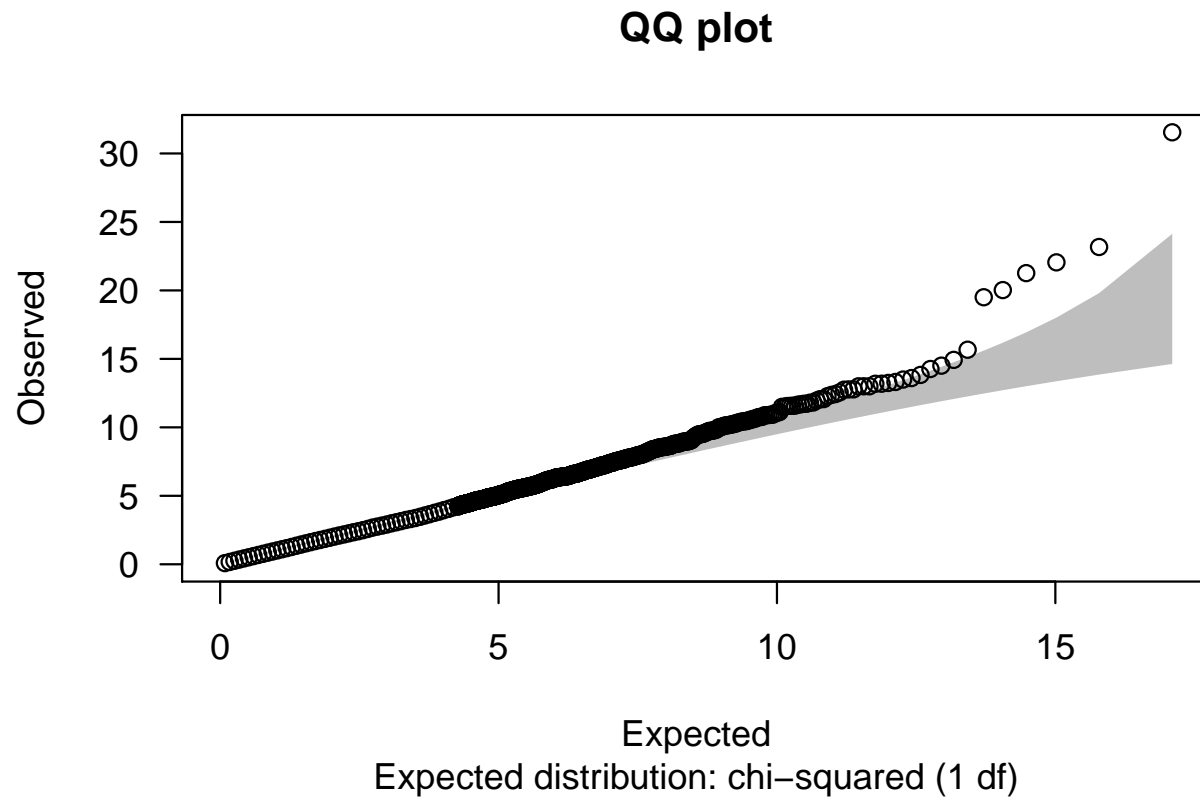
```
qq.chisq(chi.squared(tests, df=1)[use], df=1)
```

QQ plot



GWAS analysis

```
tests2 <- single.snp.tests(cc, stratum, data = subject.support,  
+ snp.data = snps.10)  
qq.chisq(chi.squared(tests2, 1)[use], 1)
```



Handling short-read sequences

The `ShortRead` package handles short-read (aka 'next-generation') sequencing reads, mostly for QC and preprocessing.

It comes with a (small) subset of a Solexa sequencing run: we will look at this example.

The data are in a structured set of folders, so the first step is to specify where they can be found.

```
exptPath <- "/Users/tlumley/Library/R/2.15/library/ShortRead/extdata"
```

and then use the `SolexaPath()` function to create a simple summary of the structure, which we will pass to other functions instead of a filename

Handling short-read sequences

```
> sp <- SolexaPath(exptPath)
> sp
class: SolexaPath
experimentPath: /Users/tlumley/Library/R/2.15/library/ShortRead/extdata
dataPath: Data
scanPath: NA
imageAnalysisPath: C1-36Firecrest
baseCallPath: Bustard
analysisPath: GERALD
> imageAnalysisPath(sp)
[1] "/Users/tlumley/Library/R/2.15/library/ShortRead/extdata/
    Data/C1-36Firecrest"
> analysisPath(sp)
[1] "/Users/tlumley/Library/R/2.15/library/ShortRead/extdata/
    Data/ C1-36Firecrest/Bustard/GERALD"
```

Reading data

The function `readAligned()` reads in the aligned sequence fragments,

```
> aln <- readAligned(sp, "s_2_export.txt")
> aln
class: AlignedRead
length: 1000 reads; width: 35 cycles
chromosome: NM NM ... chr5.fa 29:255:255
position: NA NA ... 71805980 NA
strand: NA NA ... + NA
alignQuality: NumericQuality
alignData varLabels: run lane ... filtering contig
```

Examining data

The `sread()` function returns the bases read

```
> sread(aln)
```

```
A DNASTringSet instance of length 1000
```

```
width seq
```

```
[1]    35 CCAGAGCCCCCGCTCACTCCTGAACCAGTCTCTC
```

```
[2]    35 AGCCTCCCTCTTTCTGAATATACGGCAGAGCTGTT
```

```
[3]    35 ACCAAAAACACCACATACACGAGCAACACACGTAC
```

```
[4]    35 AATCGGAAGAGCTCGTATGCCGGCTTCTGCTTGGA
```

```
[5]    35 AAAGATAAACTCTAGGCCACCTCCTCCTTCTTCTA
```

```
...
```

and the `quality()` function gives the read quality, coded with Z as the best

Examining data

```
> quality(aln)
class: SFastqQuality
quality:
  A BStringSet instance of length 1000
      width seq
[1]    35 YQMIMIMMLMMIGIGMFICMFFFIMMHHIHAAGAH
[2]    35 ZXZUYXZQYYXUZXYZYYZZXXZZIMFHXQSUPPO
[3]    35 LGDHLILLLLLLLIGFLLALDIFDILLHFIAECAE
[4]    35 JJYYIYVSYYYYYYYYSDYYWVUYNNVSVQQELQ
[5]    35 LLLILIIIDLLHLLLLLLLLLLLLLALLLHLLLEL
[6]    35 YYYYYYYWVVMGGUHQHQMUFCMCDHQHEDDD
[7]    35 ZZZZZZZZZYZZZZZYZZZZYZZZZZZZUUUUU
[8]    35 ZZZZZZZZUZZUZZZZZZZZZZZZZYZZZUUHUH
[9]    35 ZZZZZZZYZZYZZZZYZZZZZZZZZZZZZZXUNUUU
```

Note how the quality goes down later in the read.

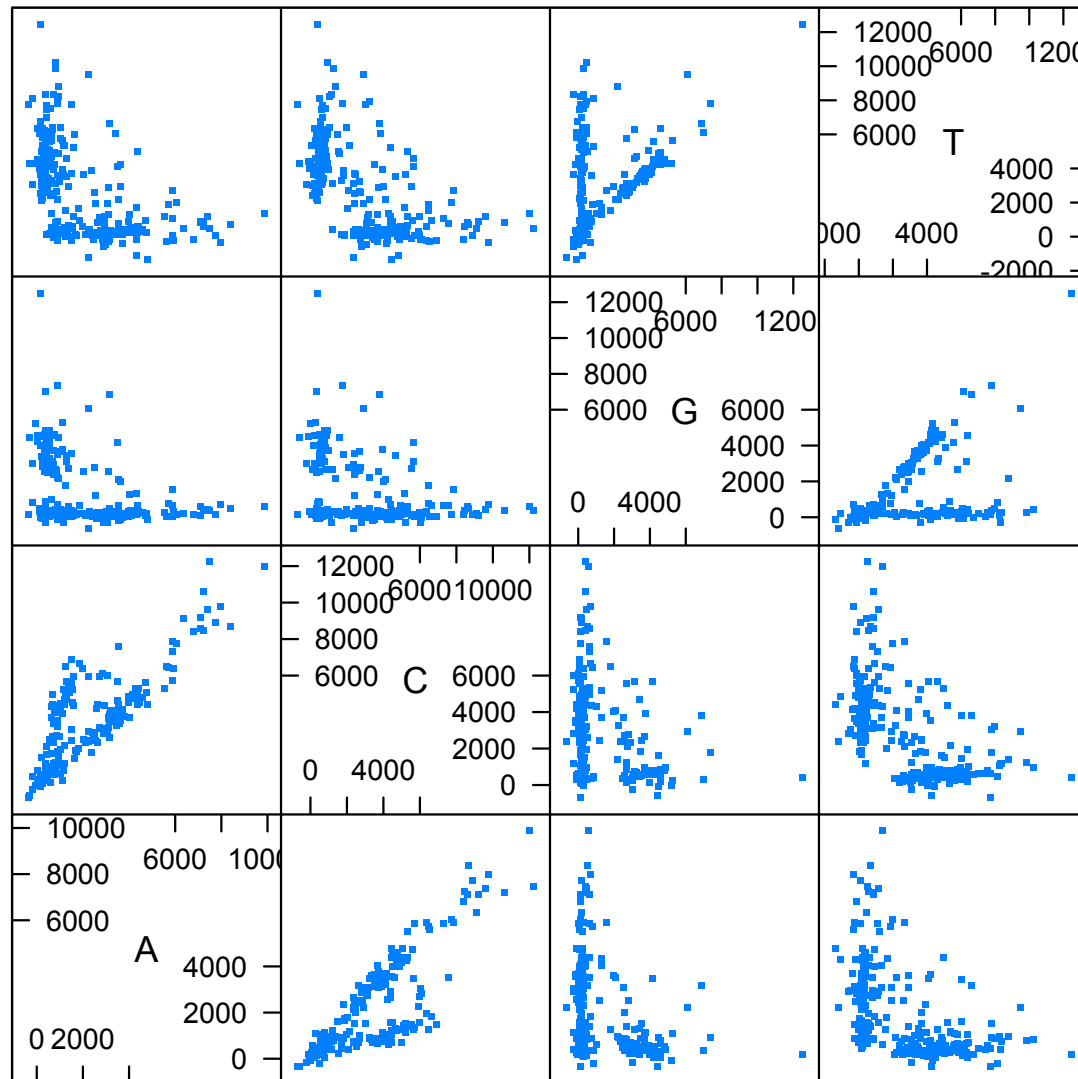
Examining data

We can read other information, such as the fluorescence intensities for each base

```
> int <- readIntensities(sp)
> int
class: SolexaIntensity
dim: 256 4 36
readInfo: SolexaIntensityInfo
intensity: ArrayIntensity
measurementError: ArrayIntensity
> splom(intensity(int)[[,2]], pch=".", cex=3)
> splom(intensity(int)[[,35]], pch=".", cex=3)
```

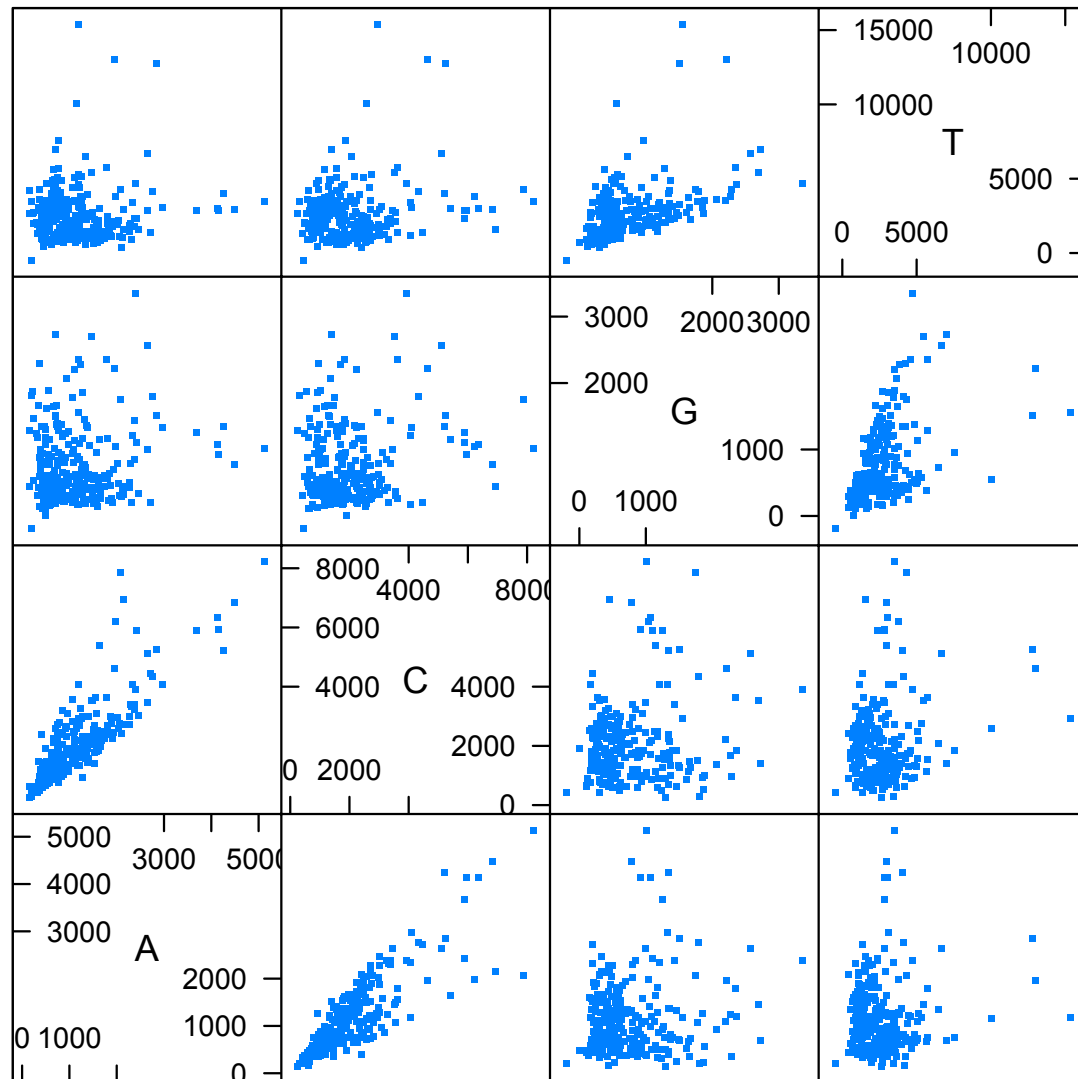
and then create a scatterplot matrix of the intensities for the 2nd and 35th bases. The plots show the correlation between A-T and C-G channels, and the deterioration towards the end of the read.

Intensities: 2nd position



Scatter Plot Matrix

Intensities: 35th position



Scatter Plot Matrix

Differential expression by RNAseq

The `edgeR` package (among others) estimates differential RNA expression from RNAseq experiments. It's a sequel to the `limma` package for microarray gene expression data.

RNAseq produces a count for each transcript, rather than the continuous measure produced by microarray experiments, and the statistical analysis relies on models for variation in counts.

We will look at an RNAseq experiment comparing gene expression in prostate cancer cells with and without treatment with an androgen (a testosterone analogue).

The experiment had three treated samples and four control samples, sequenced in seven of the lanes of a single flow cell on an Illumina 1G sequencer.

The data have been mapped to the human genome and turned into counts of transcripts in a simple text file

RNAseq of prostate cancer

```
> x <- read.delim("pnas_expression.txt", row.names=1,
  stringsAsFactors=FALSE)
> head(x)
```

	lane1	lane2	lane3	lane4	lane5	lane6	lane8	len
ENSG00000215696	0	0	0	0	0	0	0	330
ENSG00000215700	0	0	0	0	0	0	0	2370
ENSG00000215699	0	0	0	0	0	0	0	1842
ENSG00000215784	0	0	0	0	0	0	0	2393
ENSG00000212914	0	0	0	0	0	0	0	384
ENSG00000212042	0	0	0	0	0	0	0	92

The `stringsAsFactors` argument is needed because we want to keep the transcript names as strings, not turn them into factors.

RNAseq of prostate cancer

We also need to specify which 'lanes' are treated and which are control.

```
> targets
```

	Lane	Treatment	Label
Con1	1	Control	Con1
Con2	2	Control	Con2
Con3	2	Control	Con3
Con4	4	Control	Con4
DHT1	5	DHT	DHT1
DHT2	6	DHT	DHT2
DHT3	8	DHT	DHT3

RNAseq of prostate cancer

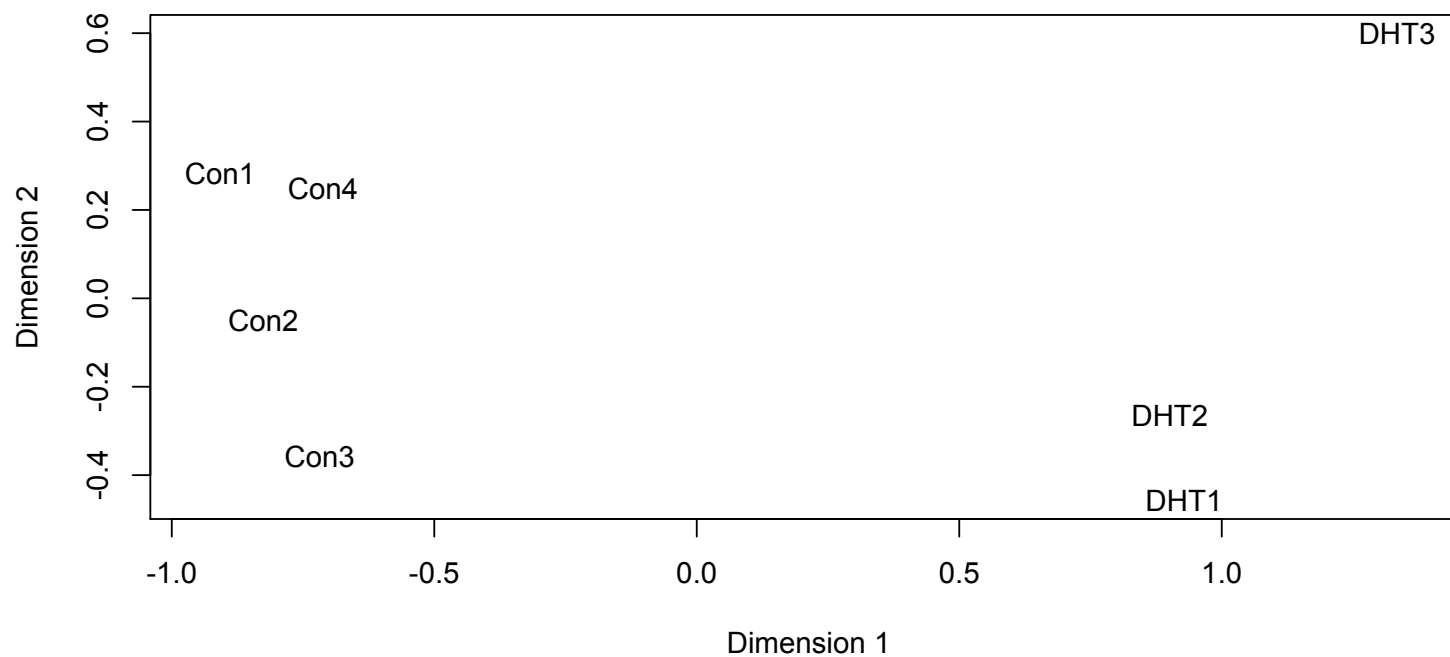
Putting the two together and filtering out low-expression transcripts gives data for analysis

```
y <- DGEList(counts=x[,1:7], group=targets$Treatment, genes=data.frame)
colnames(y) <- targets$Label
keep <- rowSums(cpm(y)>1) >= 3
y <- y[keep,]
y$samples$lib.size <- colSums(y$counts)
```

Here, `cpm()` means counts per million

Overall differences?

`plotMDS(y)` does multidimensional scaling, projecting the data into two dimensions to see how well separated the samples are. In this case there are big treated/control differences and smaller between-replicate differences.



Differential expression (at last)

The statistical analysis has two steps. First, the variability is estimated, then the treatment and control groups are compared for each gene, using a test related to Fisher's exact test.

```
y <- estimateCommonDisp(y, verbose=TRUE)
```

```
y <- estimateTagwiseDisp(y)
```

```
et <- exactTest(y)
```

```
top <- topTags(et)
```

Differential expression (at last)

The output gives the gene name the log fold-change, the overall abundance, the p -value and FDR.

Comparison of groups: DHT-Control

	Length	logFC	logCPM	PValue	FDR
ENSG00000151503	5605	5.704455	9.651026	0.000000e+00	0.000000e+00
ENSG00000096060	4093	4.881110	9.886097	1.065531e-315	8.787436e-312
ENSG00000166451	1556	4.539306	8.781549	1.168385e-217	6.423783e-214
ENSG00000127954	3919	8.054706	7.152971	3.838688e-200	1.582883e-196
ENSG00000162772	1377	3.204621	9.689874	8.819449e-170	2.909360e-166
ENSG00000113594	10078	3.960785	7.986306	1.108187e-144	3.046406e-141
ENSG00000116133	4286	3.143633	8.740383	1.291088e-138	3.042172e-135
ENSG00000115648	2920	2.513583	11.429414	6.775660e-130	1.396972e-126
ENSG00000123983	4305	3.473311	8.534717	9.250437e-129	1.695297e-125
ENSG00000116285	3076	4.104133	7.306976	1.758623e-128	2.900673e-125

Differential expression (at last)

We could use `biomaRt` or one of the other annotation packages to convert to other gene names

```
> getBM(attributes="hgnc_symbol",filters="ensembl_gene_id",  
        values=rownames(top$table),mart=human)
```

```
hgnc_symbol
```

```
1      FKBP5  
2      LIFR  
3      MLPH  
4      DHCR24  
5      ERRFI1  
6      ACSL3  
7      STEAP4  
8      NCAPD3  
9      ATF3  
10     CENPN
```